DRAFT

Full paper published in:
Reliability Engineering and System Safety,
Vol. 89 (2005), pp. 33-47

Trust Case: justifying trust in an IT solution

J. Górski[1,2], A. Jarzębowicz[1], R. Leszczyna[2], J. Miler[1], M. Olszewski[1]

[1]Gdańsk University of Technology, Narutowicza 11/12, 80-952 Gdańsk, Poland
[2]EC Joint Research Centre, IPSC, I-21020 Ispra (VA), Italy

Janusz Górski
Gdańsk University of Technology,
Narutowicza 11/12, 80-952 Gdańsk, Poland
Tel: +48 58 347 1909, fax: +48 58 347 2727, jango@pg.gda.pl

Abstract. In the paper we present an approach to the trust case development for DRIVE, the IT infrastructure supporting the processes of drugs distribution and application. The objectives of DRIVE included safer and cheaper drugs distribution and application. The *trust case* represents an argument supporting the trustworthiness of the DRIVE solution. It is decomposed into *claims* that postulate some trust related properties. Claims differ concerning their abstraction level and scope. To express a claim we need a language and a conceptual model. We used the Unified Modeling Language (UML) to represent *claim models* and the related *context models* of the trust case. To specify claims we introduced *Claim Definition Language* – CDL. The paper gives a detailed description of the above concepts and illustrates how they were applied in practice.

1. Introduction

As we are becoming more and more dependent on software (both in the individual and in the group dimensions) there is an increasing need for *software trustworthiness* understood as a guarantee that the trust that the system will meet the most critical expectations (e.g., safety and/or security) is well justified and based on evidence rather than on beliefs. This evidence can include analytical results showing that the system objectives, scope and requirements are adequately identified and understood, probabilistic failure profiles of systems and components, design decisions of which impact on safety and security is known and well understood, results of additional analyses, proofs of conformity to accepted and recommended standards and guidelines and so on.

The concept of a *trust case* refers to the need of providing a complete and explicit argument justifying trust in the computer system being used in a given application context. As the trust case encompasses both, safety and security guarantees, it could be (conceptually) split into two parts: the safety case and the security case.

Safety case is a documented body of evidence that provides a convincing and valid argument that a system is adequately safe for a given application in a given environment. It is recommended that safety case is being developed

in parallel with the design. One standard [1] stresses that "the Safety Case should be initiated at the earliest possible stage in the Safety Program so that hazards are identified and dealt with while the opportunities for their exclusion exist". The structure of safety cases has been examined by some EU sponsored projects (e.g., [2]).

Much work has been done on security evaluation of products and systems. Two emerging standards seem to be especially influential: [3] for security management and [4] (complemented by [5]) for security evaluation of products and systems. A security case is the counterpart of the safety case and collects the evidence justifying the trust in that the system is sufficiently secure.

Safety and security are two different (although not disjoint) system qualities. If both matter in a given situation (as it is true for medical applications) they can sometimes be in conflict. As an example take a patient related data that should have its access controlled and restricted due to privacy considerations (a security related requirement) and simultaneously should have high availability with relaxed access control in emergency situations (a safety related requirement). In such applications the trust case should cover both, safety and security viewpoints and in addition it should consider possible conflicts and their resolutions.

The IST-DRIVE project, performed within the 5th EU Research Framework Programme, focused on medical care, undoubtedly a trust related domain. In DRIVE both safety and security mattered and consequently both aspects had to be adequately covered. We used a more neutral concept of trust in order to start with a single objective and then to specialize it into security and safety objectives [6].

In the subsequent sections we first relate our approach to the work of others and provide a brief description of DRIVE objectives and scope. Then we introduce the trust case conceptual structure and present the language used to specify the trust case. Next we give more details on the structure and contents of the DRIVE trust case and the way we handled the evidence (facts and assumptions) within this structure. In conclusions we summarize our contribution and present plans for future research.

2. Related Work

The approach to trust cases is derived from the safety critical systems domain where the notion of *safety case* has already a well established meaning. Two most popular approaches to safety cases currently in use are: Adelard's ASCAD - Claims Arguments Evidence approach [7, 8] (motivated by the research done in [2]) and GSN - Goal Structuring Notation, proposed by the University of York [9].

In ASCAD, the safety case is structured into claims that are supported by arguments that refer to evidence. ASCAD does not propose any specialized language to express claims, arguments and their relationships except some graphical symbols. A safety case is a graph and the meaning of particular elements is given in a natural language. The process of developing and managing a safety case is supported by a set of guidelines [7] and a tool [10].

Goal Structuring Notation (GSN) provides more detailed concepts concerning the structuring of safety arguments. GSN diagrams are constructed using several types of nodes, including: *goal* (similar to "claim" in ASCAD), *strategy* (explanation of the approach to the decomposition of a particular goal), *rationale* (explanation and justification of a given strategy), *context* (explanation of the context in which a given strategy is valid), *model* (denotation of the system model a given goal and strategy refer to) and *justification* (evidence supporting the goal). The advantage is that this notation provides for defining complex arguments and allows engineers to clearly document their "way of thinking", i.e. the approach they took while decomposing a given safety goal [9].

In relation to the above approaches the one presented in this paper differs in that we introduce claim context modeling as the primary means to control the scope and the abstraction level of expressing claims. In particular, we require that each claim is associated with a context model and we recommend Unified Modeling Language (UML) [11] as the corresponding modeling language. In addition to this we introduce a formalized language to express claims that helps in increasing precision and removing possible ambiguities. As our target is to provide a framework for analyzing and justifying trust in systems rather than focusing on system safety we recognize that in many cases such trust is conditional and based on numerous assumptions without further justification. Therefore we emphasize the need

for a separate construct to represent assumptions and we are particularly focused on *assumption management*. This will be discussed in the following sections.

3. Project DRIVE – Objectives and Scope

The approach presented in the paper was applied in the European Union 5th Framework Programme R&D project DRIVE[1] (DRug In Virtual Enterprise). The project aimed at creating a safer and smarter hospital environment by means of innovative and trustworthy IT solutions. DRIVE involved ten partners from Italy, Sweden, Spain, France and Poland representing hospitals, research institutions, software and pharmaceutical companies, and hospital equipment manufacturers. The strategic objective of DRIVE was to improve the quality of patient care and patient safety by reducing the drug administration errors while simultaneously decreasing the supply chain costs. The project resulted in an integrated IT solution (hereafter called *DRIVE solution*) supporting the drug distribution, administration and monitoring processes from pharmaceutical companies through warehouses and pharmacies down to a patient in a hospital. A pilot version of the DRIVE solution was tested in the Hospitale San Raffaele in Milan, Italy.

DRIVE solution focuses on three key areas of healthcare optimization:

- Clinical process: representing the drug related processes within a hospital directly or indirectly aiming at the improvement of patient's health state, from the admission until discharge.
- Supply chain: representing the flows of the pharmaceutical products and related information, from the manufacturer to the point of use (patient's bedside).
- Trust: representing the stakeholder requirements for the protection of critical assets, such as personal clinical records, hospital professional accountability, enterprise value chain and privacy within the entire business model.

More thorough description of the project and its results can be found in [12]. In [13] an overview of the access control mechanisms implemented in DRIVE is presented. Fig. 1 shows the structure of business processes addressed by DRIVE.

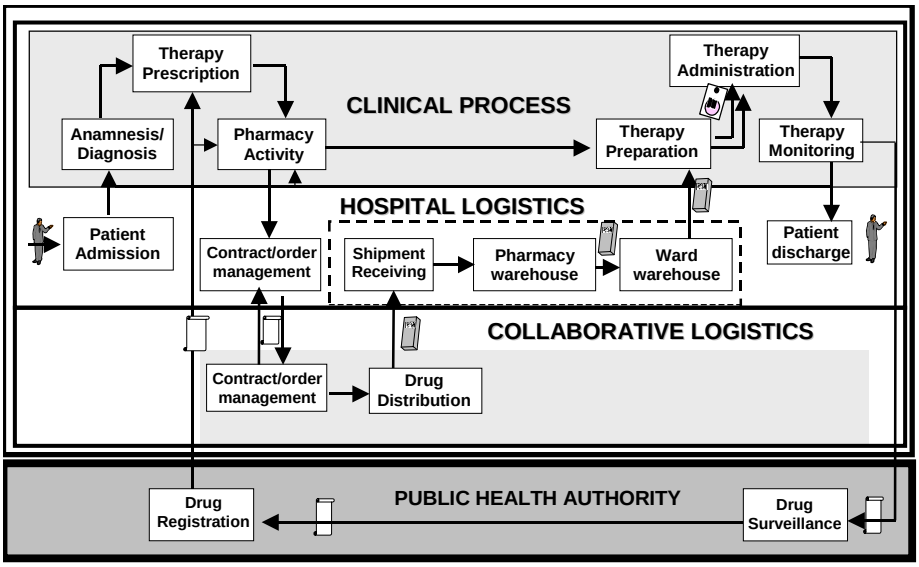

Fig. 1. The scope of DRIVE Solution.

The Clinical Process includes activities directly related to medical treatment of patients from a patient's admission to hospital until his/her discharge. The process involves gathering patient's past clinical data and diagnosis of his/her

[1] IST-1999-12040

health state (Anamnesis/Diagnosis), prescribing the adequate drug therapy (Therapy Prescription), approving and ordering the prescribed drugs (Pharmacy Activity), preparing personalized drug doses (Therapy Preparation), applying drugs to the patient (Therapy Administration) and observing the results by collecting patient's vital parameters (Therapy Monitoring). The logistics of providing drugs is divided into two processes: Collaborative Logistics which focuses on acquiring drugs through cooperation with external sources (manufacturers) and Hospital Logistics which handles drug flow through placing orders, receiving drugs delivered by manufacturers and distributing them to hospital warehouses and then to pharmacies and wards. The Public Health Authority process handles information about effects of drug application, including Adverse Drug Events (ADE). This process was not included within the scope of DRIVE. Nevertheless it was assumed that DRIVE inputs to this process the results from Therapy Monitoring and uses its outputs to validate the prescribed therapies.

All patient identification data is captured in a 2D coded wristband worn by a patient from admission to discharge. DRIVE solution supports the physician during the anamnesis, diagnosis and therapy prescription phases. It also supports nurses in therapy preparation and administration with the aim to reduce the likelihood of human errors that often occur during these phases.

The hospital logistics part of DRIVE solution supports the head nurses in maintaining the proper stock of pharmaceutical products in ward cabinets. A new 2D label, specially studied for DRIVE, encodes all the information needed for tracing the drug flow (product code, expiration date, lot number, serial number).

The collaborative logistics part of the DRIVE solution provides for sharing the logistics information between hospitals and pharmaceutical companies through a web-based infrastructure. It provides for automated exchange of price lists, orders, order confirmations, shipping notes and invoices.

A number of design decisions were related to trust, e.g., drug and wristband labels to uniquely identify drugs and patients, smart cards and PIN codes to identify and authenticate healthcare professionals, digital signatures to protect integrity of important assets and to provide for non-repudiation, role-based access control and so on.

The need of providing the *trust case*, an integrated overall argument explicitly stating the trust objectives and linking them with the supporting evidence, was recognized later in the project course. It resulted in extending the project scope by adding an additional work package devoted to the trust case development.

4. Trust Case Conceptual Model

Intuitively *trust* is a notion referring to a belief in some postulated property (hereafter called *trust objective*) like honesty, truthfulness, competence, reliability or safety of the trusted object (the *trustee*) considered in a specific *context*. The trustee is any entity perceived as important and interesting from the perspective of the trusting entity (the *trustor*). In particular, the trustor can be a part of the context within which the trustee is taken into consideration, for instance we can consider patient's safety (the postulated property) during medical procedures (the trustee) within the hospital environment (the context) from the patient's perspective (the trustor). Trust is not a binary notion (i.e., either we trust something or not), it is rather represented as a value in some *trust scale*. The actual level of trust is influenced by *trust justification* that refers to the context within which the trust is being considered. For instance, a patient's level of trust in the therapist's competence depends on the therapist's professional certificates, the recommendations from other patients, the environment within which the therapy is being applied (e.g., hospital, home, casual situation) and many others. If such justification of trust exists objectively (is explicitly documented) we call it a *trust case.* Note that we do not postulate that trust justification (the trust case) *determines* the level of trust in the claimed property. Instead, we rather postulate that a trust case *influences* the trustor's level of trust. This is in recognition of the fact that our understanding of all factors that can affect trust is far from being complete, for instance two different persons can develop different trust levels in the same thing even if presented with exactly the same argumentation (take as an example the attitudes to different religions). More details of this approach to trust can be found in [14].

A trust case is developed by making an explicit set of claims about the system and then collecting and producing supporting evidence and developing a structured argument that the evidence justifies the claims. A conceptual UML model describing the trust case elements and showing their relationships is shown in Fig. 2 (the arrows show the

direction of reading the association names). The conceptual model of our trust case is based on the results of SHIP[2] [2].

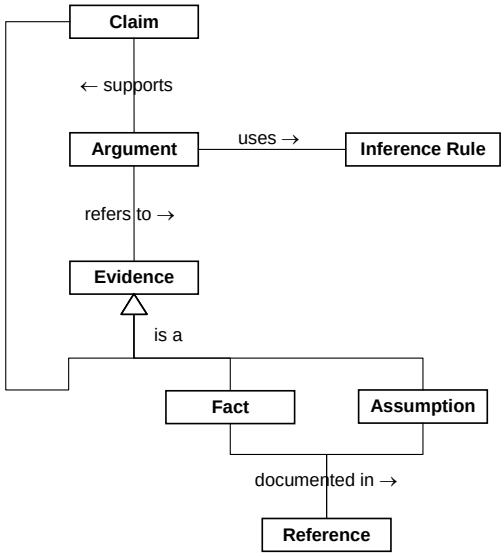

Fig. 2. The conceptual model of a trust case.

The evidence referred to by an argument for a claim can be of the following type:

- Fact – a statement or assertion of verified information about something that is the case or has happened e.g. demonstrated adherence to established principles, documented design decisions, results of performed analyses;
- Assumption – a statement assumed to be true for which we do not include any supporting material;
- Claim – a (postulated) property that is explicitly justified by giving the argument and evidence that support it.

Note that according to the above definition, claims can be recursively supported by other claims (their sub-claims) to the depth that is arbitrarily chosen by the trust case designer. We also assume that facts and assumptions are linked to references (documents, reports, data, models and so on) maintained together with the trust case.

The inference rules used in arguments are of three basic types:

- *logic* (establishing the validity of a claim from the logical assertions given by the supporting evidence),
- *probabilistic* (e.g. justifying the failure probability postulated by a claim from the probabilities given by the supporting evidence),
- *qualitative* (establishing the validity of a claim by referring to the common acceptance of good practices that were adhered to as demonstrated by the evidence supporting the claim). The qualitative argumentation can in particular refer to accepted standards, guidelines or so called "engineering judgment".

The nature of software faults differs from the nature of hardware random faults in that software is not subjected to physical degradation through aging, vibration, humidity, or dust. Software faults are design faults (caused by humans) and their statistical properties are very hard to quantify. Therefore, for software intensive systems, the role of the logical and qualitative arguments is increased at the expense of the probabilistic arguments.

As shown in Fig. 3, a trust case develops into a tree-like structure (or multi tree, in case we have multiple trust objectives) of arbitrary depth and is completed when all the leaves represent (well documented) facts and assumptions.

[2] SHIP (Assessment of the **S**afety of **H**azardous **I**ndustrial **P**rocesses in the Presence of Design Faults) – the project (ref. EV5V 103) performed within the EC framework of the ENVIRONMENT Programme, sub-theme: Major Industrial Hazards.

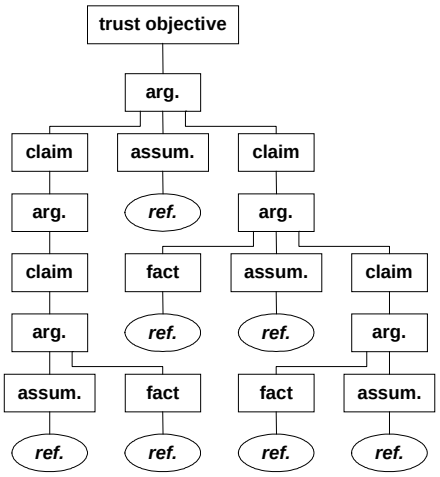

Fig. 3. The structure of a simple trust case.

As we can always convert a given sub-claim in the tree into an assumption, the depth of the tree remains under control of the trust case developer. However, in such situation the trust case is conditional and its validity becomes dependent on the validity of the assumption which is not justified within the trust case scope.

5. Modeling Claims

To define claims we need a language. A natural language (e.g., English) is the first choice, but due to its obvious limitations (the lack of precision and possible ambiguities) the natural language expressions can sometimes be misinterpreted. On the other hand, to be persuasive the trust case has to refer to the concepts of the application domain and therefore the choice of the vocabulary is limited. Another problem is that while specifying claims in a natural language it is difficult to control the scope and it is easy to mix the levels of abstraction a given claim refers to that adversely affects the clarity and soundness of the supporting argumentation.

To overcome those difficulties we decided to control the language associated with a claim by introducing what we call the Claim Definition Language (CDL). CDL introduces the following means that help the analyst to be more precise and unambiguous while building a trust case:

- A graphical language that provides constructs to represent claims, arguments, facts and assumptions and a labeling scheme that helps with their unambiguous identification. The identifier indicates the position of a given construct within the overall trust case. Each claim is associated with its *claim model* presenting a given claim, its supporting argument and all the evidence the argument refers to in a direct way.
- A graphical language that provides for representing, for each claim, its associated *context model* showing all the (physical and logical) objects that are referred to in the claim together with their relationships. To provide for avoiding possible ambiguities we also made some steps towards formalization of the meaning of the relationships occurring in context models.

The constructs used to define claim models are shown in Fig. 4.

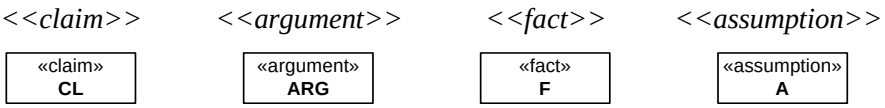

Fig. 4. Elements of the claim model.

All the elements of the claim model are defined as UML stereotypes. This helped us in using a UML-conscious graphical tool ([15] in our case) to maintain the trust case. An example claim model is shown in Fig. 5.

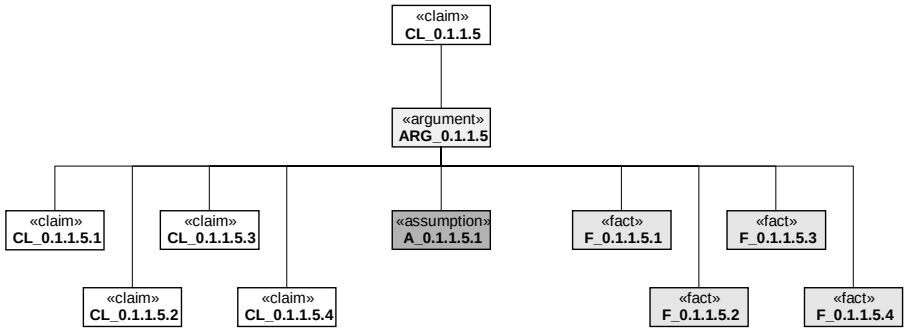

Fig. 5. An example of a claim model.

Different background colors distinguish different types of elements (claims, facts, assumptions and arguments). Each model element has its unique label that consists of its type identifier ('CL' for claim, 'ARG' for argument, 'A' for assumption, 'F' for fact) and a numeric identifier. The numeric identifier represents the element's position within the whole trust case. The example given in Fig. 5 represents the model of the claim CL_0.1.1.5. Such higher-level claim is called a *super-claim* and the claims that are referred to in the associated argument are called *sub-claims*. The argument has the same number as the claim it supports. The identifying numbers of sub-claims, assumptions and facts supporting the super-claim are constructed in such a way that they use the super-claim's numeric identifier as the prefix extended by the ordering number of this element (from left to the right) separated by a dot. For instance, CL_0.1.1.5.3 is the third sub-claim of super-claim CL_0.1.1.5 and F_0.1.1.5.2 is the second fact supporting CL_0.1.1.5.

In general, claims are structured in two dimensions: vertically (refined claims used by the arguments justifying the higher level claims) and horizontally (complementary claims that together are used in the same argument justifying a higher level claim).

6. Modeling Claim Contexts

We defined a graphical language that provides means to grasp and represent the context within which a given claim model is being interpreted. The model is expressed in terms of UML. Object orientation and UML are becoming de-facto standard in software development. One of the advantages is that they constitute conceptual and linguistic means that can be applied at the system as well as at the software levels, providing for bridging the gap between the two views. The recent attempts to extend UML to business processes (e.g. [16]) provide for covering both, the artifact being developed (the computer system and its software) and the target environment within which it is being used. Such uniform framework can be applied not only to model the system and its target environment but also to model other important processes the trust depends on, like the development process, maintenance, installation and so on. Our claim context models adhere to the common UML style and therefore can be applied to represent a broad range of business processes that can be expressed in UML. This puts the analytical task of trust case development within a uniform modeling framework.

In addition to the standard modeling means offered by UML we introduced a set of class stereotypes to represent typical objects that occurred while building our claim context models. They were especially useful while we were working with the higher level claims that were not supported by the models already developed during the system construction. Those extensions are shown in Fig. 6.

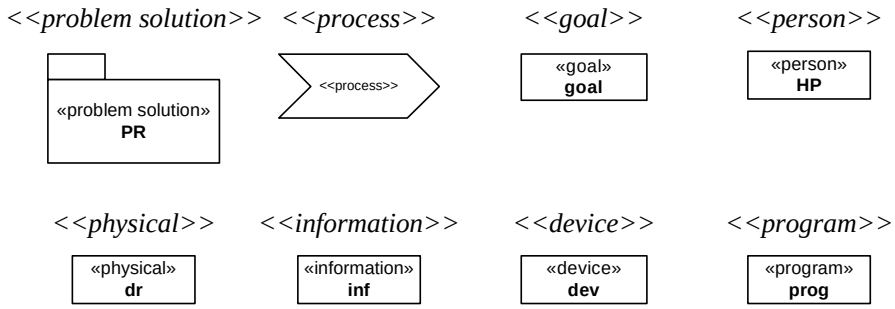

Fig. 6. Stereotypes used in claim context models.

An example claim context model taken from the DRIVE trust case [17] is given in Fig. 7. The context model has been derived from the documentation resulting from the DRIVE development process and shows the relations between patients, drugs, their physical identifiers and the corresponding logical entities. The model presents the object classes and their relationships that provide the conceptual and linguistic context used while expressing the following claim:

CL_0.1.1.6
*Drug Labels of the Drugs **applied** to Patient are **consistent** with the Prescription Data in the **corresponding** Patient Record **identified** by the Patient's Wristband Label.*

Note that the context model in Fig. 7 includes all the objects referred to in the corresponding claim. The words given in bold in the claim definition distinguish those relationships in the claim context model the claim refers to. If any of these relationships is not present in the claim context model, it is interpreted in terms of the relationships given explicitly (is a derived relationship).

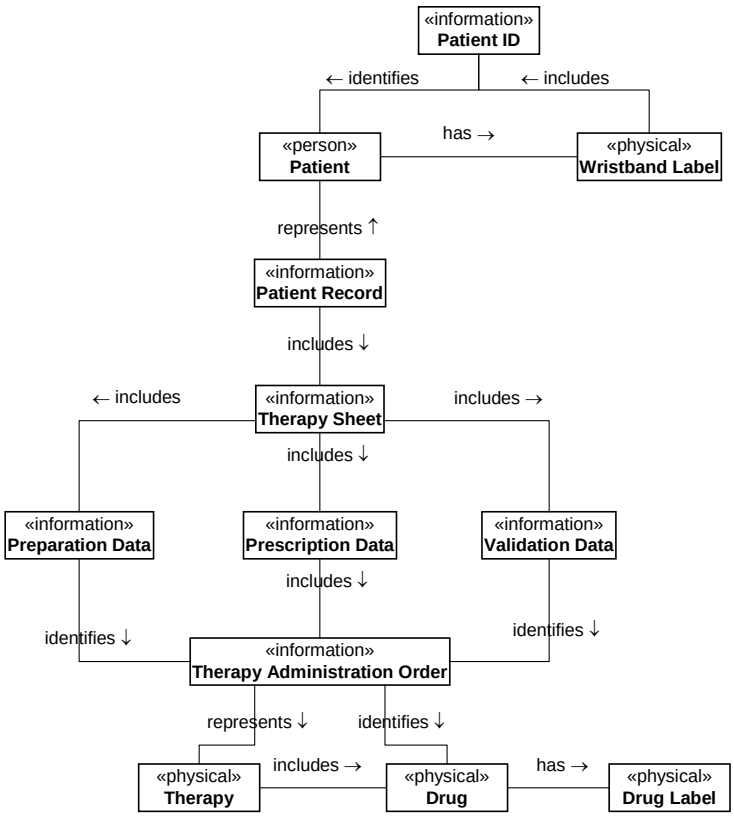

Fig. 7. Example of a claim context model.

Although the objects referred to in the claim are precisely specified in the corresponding claim context model, the relationships the claim refers to can lead to ambiguous interpretations, e.g. the meaning of: "drug *applied* to patient". This problem can be mitigated by formalizing the language.

7. Formalizing the Language

Claims, arguments, assumptions and facts were specified in a natural language referring to the elements introduced by the associated claim context model. However, such natural language specifications can sometimes give rise to ambiguous interpretations. To provide for unambiguous interpretation of claims and assumptions we made some steps towards formalizing the meaning of the basic operations that can be performed on objects and the relationships among the objects represented in a claim context model. Then we followed a convention that to help to resolve possible misinterpretations, for each claim, its natural language specification is supplemented by the specification given in the formalized language. An example of such formalized specification of claim CL_0.1.1.6 (see the previous section) is given in Fig. 8 (the expressions listed in new lines are supposed to be connected by logical AND).

forall
(d: Drug
p: Patient):
if
d is applied to p
then
exists
(dl: Drug Label
tao: Therapy Administration Order
pd: Prescription Data
ts: Therapy Sheet
pr: Patient Record
pwl: Patient Wristband Label):
d 1:1 dl
tao identifies dl
pr includes ts includes pd includes tao
pr represents p
pwl 1:1 p
pwl u-identifies pr
endif

Fig. 8. Formal specification of claim CL_0.1.1.6.

This specification imposes restrictions on each object set being an instance of the context model shown in Fig. 7. It explicitly refers to objects of the classes given in the context model and refers to the terms (underlined in the above text) that are explicitly defined in the *CDL Dictionary*.

CDL Dictionary contains formalized definitions of various kinds of possible dependencies between objects. CDL uses basic mathematical concepts (from logic and set theory) as well as concepts from object orientation (e.g., object, state, attribute, identity). Below we recall from the CDL Dictionary the definitions of those terms that occur in the above formalized specification of CL_0.1.1.6.

Fig. 9 defines the meaning of 'drug is applied to patient'. The definition is given by the UML sequence diagram shown in the picture. The diagram explains that if patient p takes a dose c of drug d, its state is eventually changed (as

a function of the accepted dose) and the drug is destroyed (consumed) which is depicted by the cross on the lifeline of d.

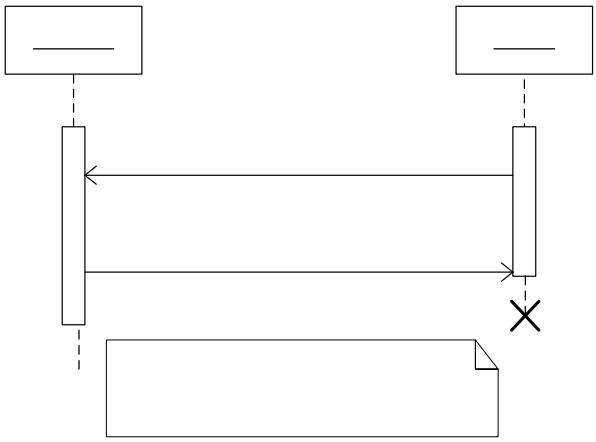

Fig. 9. UML sequence diagram defining the meaning of 'drug is applied to patient'.

The dependencies between objects that are referred to in the specification in Fig.8 are defined in Fig.10 (the symbol "==" stands for 'is defined as').

O1 designates O2 == Identity(O2) can be directly derived from Attributes(O1) and
if exists O3 such that Class(O2) = Class(O3) and O1 designates O3
then Identity(O3) = Identity(O2).
O1 1:1 O2 == O1 designates O2 and O2 designates O1.
O1 includes O2 == O2 is a part of O1.
O1 represents O2 == attributes and their values of O2 overlap with the state of O1.
O1 identifies O2 == Identity(O2) can be derived from Attributes(O1).
O1 u-identifies O2 == O1 identifies O2 and if exists O3 such that O1 identifies O3 and
Class(O2) = Class(O3) then O3 = O2.

Fig. 10. Definitions of dependencies between objects referred in Fig. 8.

According to definitions given in figures 9 and 10, the claim specification given in Fig. 8 reads as follows: if a drug is applied to a patient (which means, affects the patient's state and is consumed), the drug must have been pointed to (by means of its drug label) from the therapy administration order which was included in the prescription data which in turn was included in the patient's record representing the patient who was designated by his/her wristband label which identified the patient in the unique way.

8. Trust Case Objectives for DRIVE

The trust case for DRIVE was developed in a top-down manner. We started with the top level claim CL_0 that postulates that the DRIVE solution is trustworthy. The model that shows this topmost claim is shown in **Błąd! Nie można odnaleźć źródła odwołania.**11.

8.1. The Topmost Claim

CL_0
The DRIVE solution is trustworthy in its intended context.

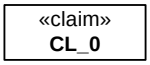

Fig. 11. Top level claim model (the model for the whole Trust Case).

Figure 12 illustrates the most general context model of the trust case, which represents the whole scope within which trust is being analyzed.

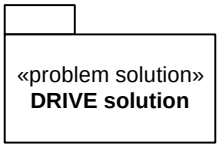

Fig. 12. Context model related to the top-level claim of the DRIVE trust case.

The model encompasses *DRIVE Solution* - the set of processes that together define the solution to the drug distribution and application problem as proposed by DRIVE.

In our work we further restricted the scope and considered only the process of using the architecture proposed by DRIVE in its intended context without taking into account the quality of the development process that resulted in this architecture, the product installation in its target environment and its further maintenance. Although we recognize that the trust depends strongly on all those processes, the DRIVE project did not address those issues explicitly and consequently it did not produce enough evidence in those aspects.

The topmost claim CL_0 was then decomposed into more specific trust objectives which resulted in the claim model shown in Fig. 13.

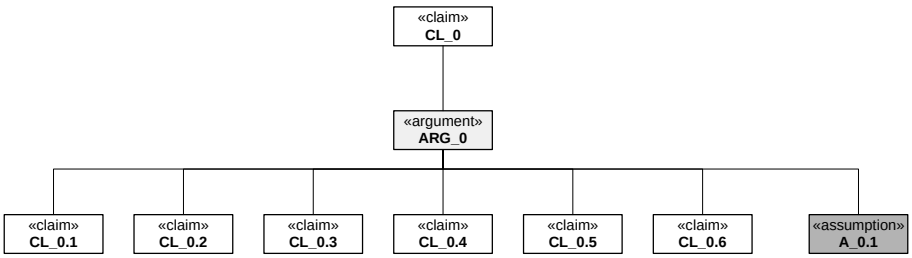

Fig. 13. Claim model for CL_0.

The associated claim context model is shown in Fig. 14. The dashed lines represent 'dependency' relationships which are used in UML business modeling extension to visualize interrelationships between processes and resources.

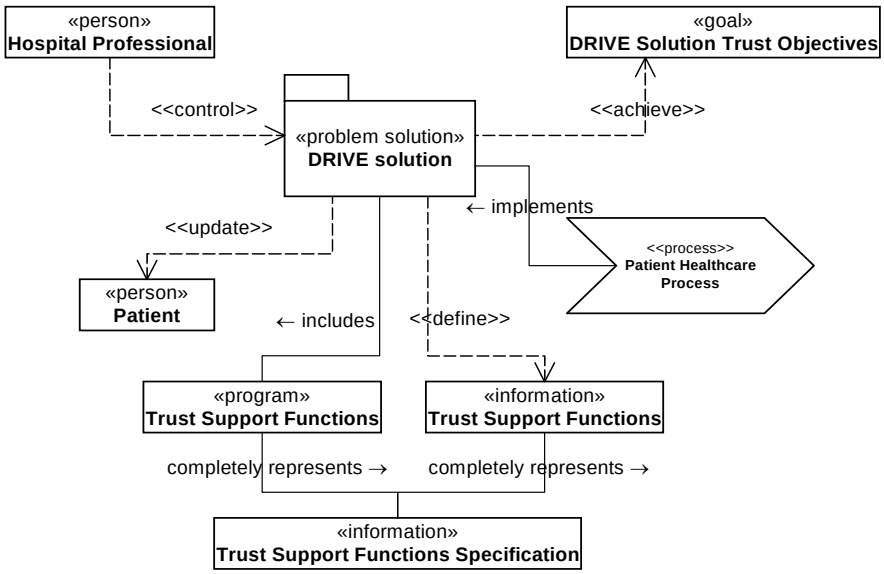

Fig. 14. Refined context model.

The context introduces the following entities:

- *Hospital Professional* – Hospital Professional involved in DRIVE Solution,
- *Patient* – Patient subjected to the drug application resulting from DRIVE Solution,
- *Patient Healthcare Process* – a process implementing DRIVE Solution,
- *DRIVE Solution Trust Objectives* - a set of trust objectives concerning safety and security as perceived by DRIVE Solution stakeholders,
- *Trust Support Functions* – set of functions to support safety and security of stakeholders,
- *Trust Support Functions Specification* – functional and design specification of Trust Support Functions.

The argument associated with CL_0 has the form:

ARG_0
Based on the A_0.1, if the claims CL_0.1 to CL_0.5 are justified and in addition are contradiction-free (CL_0.6), CL_0 is justified as well.

and is based on the following assumption:

A_0.1
The analysis of trust objectives in DRIVE is complete and valid.

This assumption reflects the fact that the trust objectives were predefined by the DRIVE project and no further analyses were performed to identify their completeness and validity (and consequently there was no further evidence available). Nevertheless, A_0.1 reflects this explicitly in the trust case and could be later converted into a claim, provided further evidence were produced, for example, by performing additional analyses of stakeholders needs.
The claim model includes the following sub-claims:

CL_0.1
DRIVE solution maintains Patient's safety.

CL_0.2
DRIVE solution maintains privacy of Hospital Professional.

CL_0.3
DRIVE solution maintains Patient's privacy.

CL_0.4
DRIVE solution ensures that all actions related to Patient's health are accountable.

CL_0.5
Trust support functions used in DRIVE are reliable.

CL_0.6
The claims CL_0.1 – CL_0.5 are mutually consistent.

The claim CL_0.6 recognizes the need for additional argumentation that the remaining claims do not contradict each other. An example of such conflict could be when the provisions for patient's privacy compromise his/her safety, for instance, by restricting access to patient's health data in case of emergency.

In the subsequent steps the above claims were decomposed into more specific trust objectives which resulted in the corresponding claim models. Due to the space limitations, in the following subsections we provide more details on just one of DRIVE trust objectives: the patient's safety. More details can be found in DRIVE deliverables [17].

8.2. Patient Safety

The patient's safety trust objective CL_0.1 was decomposed further which resulted in the claim model shown in Fig. 15.

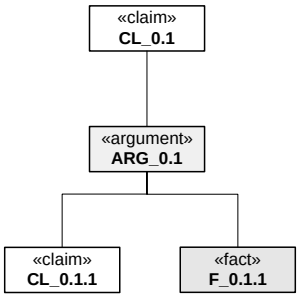

Fig. 15. Claim model for CL_0.1.

The argument associated with CL_0.1 has the form:

ARG_0.1
The scope of DRIVE is restricted to drug application and does not include the quality of medical treatment applied to a patient and the resulting consequences to the patient's health. In such case CL_0.1.1 together with F_0.1.1 are enough to ensure patient's safety.

The claim context model is documented in the following fact:

F_0.1.1
From the DRIVE documentation, the basic objects related to patient's safety and their relationships can be represented as in the context model in Fig. 16.

This context model was common for all trust objectives (Patient Safety, Hospital Professional Privacy, Patient Privacy, Accountability) that were considered in the DRIVE trust case.

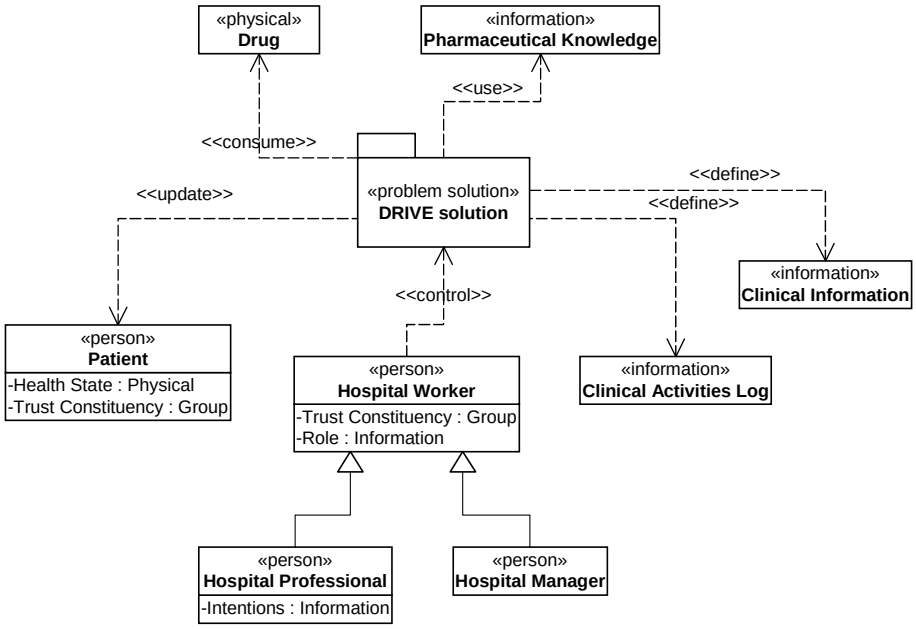

Fig. 16. Context model for CL_0.1.1.

The context model introduces the following entities:

- *Hospital Worker* – generic worker of a hospital running healthcare processes as defined in DRIVE solution,
- *Hospital Worker* $_{Trust\ Constituency/Role}$ - Hospital Worker inside certain Trust Constituency in certain Role[3],
- *Hospital Manager* – Hospital Worker with managerial responsibilities, not involved directly in clinical activities on Patients,
- *Drug* – pharmaceutical product with defined pharmacological properties,
- *Clinical Activities Log* – log of decisions and activities carried out by Hospital Professionals concerning Patients,
- *Clinical Information* – information on Patient acquired and used in DRIVE Solution for clinical needs,
- *Patient* $_{Trust\ Constituency}$ – Patient (person) inside certain Trust Constituency,
- *Patient* $_{Health\ State}$ – actual state of Patient's health while being treated inside DRIVE Solution,
- *Pharmaceutical Knowledge* – entire knowledge related to therapeutic properties of pharmaceuticals.

The super-claim CL_0.1 is supported by the following sub-claim:

CL 0.1.1
If Patient receives Drug then it is the result of a prescription by Hospital Professional who knows the actual health state of Patient and Drug actually applied to Patient is consistent (type, dose, time) with the intentions of Hospital Professional and with Pharmaceutical Knowledge.

9. Trust Case for DRIVE

The trust objectives were then further decomposed in the search for the supporting evidence. Our objective was to develop an architectural trust case, which means that we restricted the scope to the design decisions made during the development process and to their proper implementation and deployment in the given context. As the decomposition process progressed, we arrived at more specific claims referring to the architectural components, such as particular

[3] The concepts of *Trust Constituency* and *Role* are explained in [13].

workstations with associated services and particular functionalities representing business logic rules. Some of the lower level claims postulated a reliable implementation of particular components of the architecture. Those claims were finally closed by converting them into assumptions (we did not have access to the evidence supporting them).
There were two main sources of the evidence referred to in the trust case:

- Already existing evidence that was *discovered* by means of studying the DRIVE design documentation and interviewing the DRIVE design team;
- New evidence that was *produced* by performing additional analyses[4].

All claims of the trust case were specified in a natural language and supplemented with their formalized specification. This proved to be very useful while reading and reviewing the trust case as the natural language specifications suffer the obvious limitations concerning their unambiguous interpretation.
The data summarized in Table 1 gives the overview of the DRIVE trust case size (the number of arguments differs from the number of claims because due to the limited resources, not all trust case objectives were addressed in a complete way).

Table 1. Statistics of Trust Case for DRIVE.

Trust case element's name	Number of elements used
Claim	97
Argument	92
Fact	234
Assumption	121

The relatively large number of assumptions present in the trust case demonstrates that in many cases we could not find enough facts supporting claims and instead had to make assumptions. One of the reasons for this was that the trust case development activity was not included in the project from its very beginning but rather added later in the project course, when most of the development had already been done. In many cases the development activities did not produce enough evidence to support particular claims and it was often impossible to establish such evidence *post factum*. Nevertheless, it should be stressed that assumptions are rather unavoidable in case of systems which are highly dependent on humans (like DRIVE), for instance, in a situation when trust depends on intentions of the people involved.

The trust case graphical structure including claim models and claim context models was maintained in a tool. The tool, Microsoft Visio [15], supports UML and provides the stereotyping mechanism which was extensively used to support the Claim Definition Language notation. An example window from the tool showing the DRIVE trust case structure is shown in Fig. 17.

[4] For instance, some additional evidence was obtained by applying dedicated analytical techniques to selected design documents of DRIVE. The results of those analyses were then included into the trust case as additional facts.

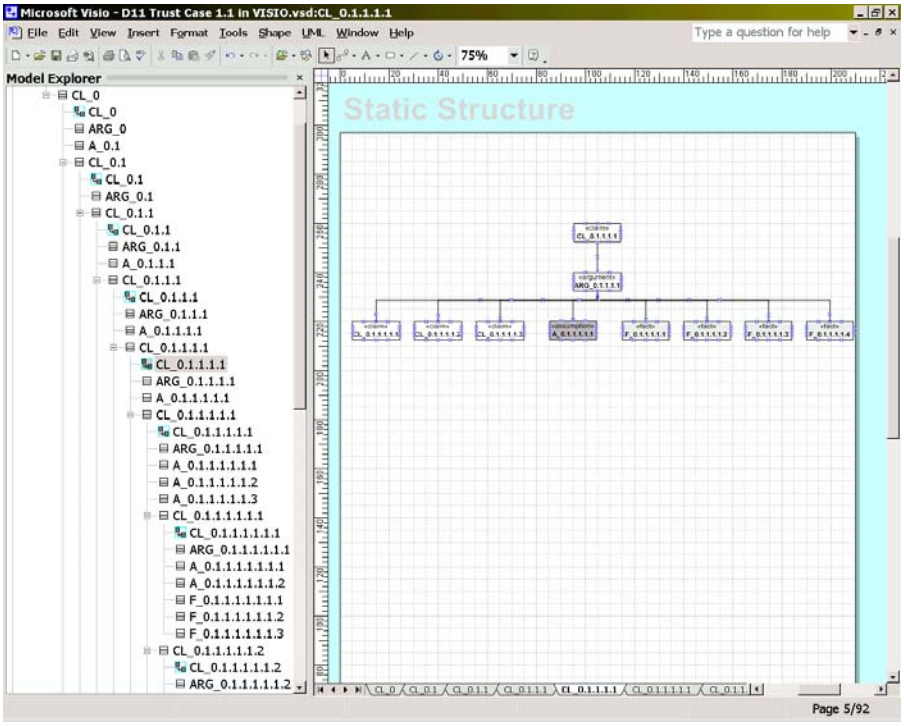

Fig. 17. A Visio 2002 window with the DRIVE trust case.

10. Facts

As we focused on the architectural and design aspects, the primary source of evidence referred to in the trust case was the design documentation of DRIVE. In most cases this evidence comprised the design decisions (facts) that were extracted from the documentation and included in the trust case. To provide for manageability of these multiple sources of data the following procedure has been applied:

- each fact represented in the main trust case document was uniquely identified and characterized by a concise textual expression that included a reference to the *fact body*,
- fact bodies (the essential quotations from the DRIVE documentation) were kept in a separate document.

An example of fact documentation is given in Fig. 18.

Fact identifier:	**F_0.1.1.1.1.3.1**
Fact textual description:	Patient ID is stored in the DRIVE Clinical Management System as part of the Patient's Record.
Fact body reference:	[D2.1 Annex 2 – Patient Record Details]

Fig. 18. The expression of a fact.

Keeping the fact bodies in a separate place helped in their review and analysis. Such reviews resulted in discovering a number of inconsistencies that were documented and then resolved during the question and answer sessions arranged with the system developers.

Due to the project constraints we could not invest much in facts generation and we primarily focused on facts discovery. Nevertheless we performed some experiments with the UML-HAZOP analytical method [18] targeting at increasing the confidence in the design decisions of DRIVE. To illustrate how this new evidence was incorporated into the trust case let us consider the claim model shown in Fig. 19. CL_0.1.1.3.5 was originally an assumption and was converted into claim after an additional evidence has been produced that is represented by the fact F_0.1.1.3.5.2.

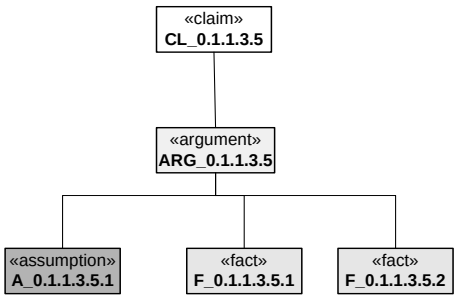

Fig. 19. Incorporation of UML-HAZOP analysis results.

The specification of the elements shown in Fig. 19 is given below.

CL_0.1.1.3.5
The structure of Patient Record provides for adequate representation of data related to Patient's Health State and Hospital Professional Intentions and all meaningful relationships between those data.

ARG_0.1.1.3.5
The validity of the structure of Patient Record was analyzed applying the UML-HAZOP technique to the class diagram representing Patient Record. The result of that analysis is a list of defects that were discovered in this diagram. Under the assumption that all of these defects were eliminated the structure of Patient Record is considered as valid.

F_0.1.1.3.5.1
Patient Record class diagram.
[Patient Record diagram derived from the DRIVE documentation]

F_0.1.1.3.5.2
UML-HAZOP analysis report with the list of detected defects.
[UML-HAZOP analysis results]

A_0.1.1.3.5.1
All detected defects were removed from the Patient Record structure.

A similar approach can be applied with respect to other evidence generating methods, for instance we are presently experimenting with incorporating into the trust case structure the evidence resulting from the formal modeling and analysis of security protocols.

11. Assumptions

In the course of trust case development we discovered a number of assumptions that were underpinning trustworthiness of the DRIVE solution. Trust context models appeared to be very helpful in identifying (sometimes hidden) assumptions that conditioned the validity of the claims. As an example take:

A_0.1.1.1
Hospital Professionals are fully qualified and their decisions are consistent with Pharmaceutical Knowledge and Patient's health state.

Without this assumption we could not argue that the DRIVE solution supports patient's safety as a wrong (intentional or unintentional) drug prescription generally could not be prevented. To express this assumption, however, we needed to refer to the concept of Pharmaceutical Knowledge that was introduced in the corresponding context model.

We consider assumptions very important components of our trust cases. Especially in situations where the system under consideration is not a safety critical one and its trustworthiness not necessarily has to be based on 'hard' evidence (e.g., take e-commerce applications) [19]. For such systems we can accept a weaker trust justification. What seems to be important however is that we are presented a complete and comprehensive picture that shows if and in which respects we are dependent on assumptions and what are their mutual relationships.

Table 2 shows the main classes of assumptions included in the trust case of DRIVE[5].

Table 2. Assumption classes in the DRIVE trust case.

Name of the class	Description	No. of assumptions
Human related	Assume certain features or behaviour of humans	9
Human-computer interaction related	Assume certain man-machine interaction pattern	7
Procedure/process related	Assume certain patterns of events, pre-post event conditions, event triggering conditions	15
Software related	Assume certain structural and/or behavioural properties of software	53
Hardware/device related	Assume certain structural and/or behavioural properties of hardware	22
Input data/entities related	Assume certain state of input data or entities that were referred to but defined outside of the trust case scope	11

Table 3 contains example assumptions of different classes extracted from the DRIVE trust case.

[5] There were 4 more assumptions of the classes that are not included in the table.

Table 3. Examples of assumptions.

Name of the class	Example assumption
Human related	A_0.1.1.3.1.1 *During Patient Admission, Admission Clerk correctly obtains Patient's Personal Data*.
Human-computer interaction related	A_0.1.1.3.1.1.1 *Data entered by Admission Office Clerk concerning Patient's Personal Data is valid*.
Procedure/process related	A_0.1.1.6.6.1 *Nurse aligns the databases of DRIVE Clinical Management Server and SmartCart before using the SmartCart off-line*.
Software related	A_0.1.1.1.1.2.2.3 *Hospital Legacy System behaves according to its specification.*
Hardware/device related	A_0.1.1.1.1.1.2.2 *The Laser Label Printer installed at SmartCart reliably prints the Patient ID included in the Patient's Wristband Label.*
Input data/entities related	A_0.1.1.2.2 *Rules Repository contains all known drug usage restrictions (reflects the present state of the Pharmaceutical Knowledge).*

The DRIVE trust case tree was developed into 7 levels. We have observed that the assumptions of the type 'human related' and 'input data/entities related' were rather evenly distributed, the assumptions of the type 'procedure/process related' were situated relatively high in the trust case structure, whereas the assumptions of the type 'human-computer interaction related', 'software related' and 'hardware/device related' tended to group in the lower levels of the tree.

12. Conclusions

The paper presents an approach used while developing a trust case for a complex IT infrastructure supporting drug distribution and application processes. The primary decision in this approach was to choose UML as the modeling framework for the trust case itself and for both, the subject domain and the solution domain under consideration. The concept of *claim context model* provided for controlling the scope and the language associated with a claim and made the trust case easy to communicate. UML with its mechanism of stereotypes proved to be a flexible and powerful tool to capture the contexts, especially those related to the higher level claims where the models had to cover a considerably broad scope, including people and physical objects involved as well as the pharmaceutical rules and knowledge. Those higher-level models were not built during system design and their development was a part of the work on the trust case. The models proved to be very useful in controlling the scope and providing the terminology for the corresponding claims.

We observed that when the trust case developed into more specific claims we could integrate into the related context models the models that were already developed during the design process. This way the analytical work smoothly incorporated the results of the analyses already performed during system design.

Another contribution of our approach was a step towards formalization of the claim definition language (CDL). Having claims and assumptions specified formally (in addition to the natural language specifications) was very useful during communication within the team (different parts of the trust case were developed in parallel and then the results were merged during the *composition meetings*) and was also confirmed during an independent review of the whole trust case.

Having UML as the underlying framework, using claim context models and formalizing the Claim Definition Language are the main contributions of our approach while comparing with the work of others [7, 8, 9, 20].

Another aspect where our work differs from what was presented in the literature is the scope of the analytical work. In our work we concentrated on the notion of *trust* which covered both safety and security (privacy, accountability) aspects. The resulting trust case brings into the surface all the facts and assumptions that support the argumentation for the trustworthiness of the analyzed target. It forms a sort of a map that shows strong and weak points of this argumentation. In our future work we consider using different symbols to distinguish the arguments of different strength to provide for easier analysis. We are also looking for tools that would support further analyses of the information included in the trust case (e.g., Dempster-Shaffer theory of evidence).

Our work was restricted in two aspects. Firstly, we concentrated our attention only on the DRIVE architecture and its usage context without taking into account other aspects that could affect trustworthiness, like development process, maintenance, installation and so on. The reason was that DRIVE was a R&D project with main focus on system architecture and design, hence it did not generate enough evidence to cover the other aspects. Consequently, our trust case is conditional and based on the assumptions that the trustworthiness of DRIVE solution is not compromised by those additional aspects. The second restriction was that the work on the trust case started late in the project, when the product was already in the implementation phase. Due to this fact, our work was a sort of *a posteriori* analysis without much influence on the system design process. An example of the design change that could have resulted from the trust case (being it developed in parallel with the design process) is an explicit verification step of a newly printed patient's 2D wristband label as not enough evidence could have been found to support the claim that the associated failure rate was acceptably low.

We have also performed some additional analyses aiming at producing more evidence for our trust case. This was done with the help of the UML-HAZOP method [18] and a related tool [21].

Visio 2002 [15] proved to be a useful tool in maintaining the trust case and helping to perform some simple consistency checks. The configuration management related to our trust case comprised of: claim models and claim context models maintained in the tool [15] and CDL specifications, claim, argument, assumption and fact specifications and the references (derived from the DRIVE documentation) maintained in files.

We plan to continue this work in both, research and experimentation. Some of the issues to be addressed include: defining a process model of trust case development (in particular, focusing on teamwork), broadening the scope of the trust case, investigating possible feedback from the trust case to the development process, addressing internal consistency of the trust case in case of potentially conflicting trust targets, getting deeper insight of the differences in the strength of the arguments, depending on the "weight" of the supporting evidence and the nature of the argument itself. Of particular interest is also to investigate how trust cases could be used to support communication, negotiation and review between stakeholders and the means to win user acceptance [19]. We also plan to investigate the possibility of better tool support and some work in this direction is already in progress.

References

[1] Defence Standard 00-56, http://wheelie.tees.ac.uk/hazop/html/56.htm

[2] SHIP – Assessment of the Safety of Hazardous Industrial Processes in the Presence of Design Faults http://www.adelard.co.uk/research/ship.htm

[3] ISO/IEC 17799:2000 Information technology - Code of practice for information security management.

[4] Common Criteria for Information Technology Security Evaluation version 2.1, 1999 (Parts 1,2,3).

[5] Common Methodology for Information Technology Security Evaluation, version 1.0, 1999.

[6] Górski J, Jarzębowicz A, Leszczyna R, Miler J, Olszewski M. An approach to trust case development, Proc. Safecomp Conference, LNCS 2788, Springer-Verlag, 2003.

[7] Adelard Safety Case Development Manual, Adelard, Adelard, London, 1998.

[8] Bishop P G, Bloomfield R E. A Methodology for Safety Case Development, Proc. Safety-critical Systems Symposium, Birmingham, UK, February 1998.

[9] Kelly T. Arguing Safety A Systematic Approach to Managing Safety Cases (1998). PhD Thesis, University of York, UK, YCST 99/05, 1998, available at http://www.cs.york.ac.uk/ftpdir/reports/YCST-99-05.ps.gz

[10] ASCE (Adelard Safety Case Editor) homepage: http://www.adelard.com/software/asce

[11] Unified Modeling Language (UML), Version 1.5, http://www.omg.org/technology/

[12] IST-Drive site: http://www.sanraffaele.org/Drive/

[13] Wilikens M, Masera M, Feriti S, Sanna A. A context-related authorization and access control method based on RBAC: a case study from the health care domain, Proc. SACMAT 2002, June 3-4, 2002, Naval Postgraduate School, Monterey, USA. ACM Press, pp 117-124.

[14] Gorski J. A Framework for Analyzing Trust in IT Systems, Proc. PSAM7&ESREL4 Conference, Berlin, June 14-18, 2004 (to appear in Springer-Verlag)

[15] Microsoft Visio 2002 Professional, 2002, http://www.microsoft.com/office/visio/

[16] Eriksson H-E, Penker M. Business Modeling with UML, J. Wiley, 2000.

[17] DRIVE D11.1-3 – Trust Case for DRIVE, D11.1-3, version 1.1, January 2003.

[18] Górski J, Jarzębowicz A. Detecting defects in object-oriented diagrams using UML-HAZOP, Found. of Comp. and Decision Sciences, vol. 27, no. 4, 2002.

[19] Górski J. How can we justify trust in software based systems?, Proc. Advanced Computer Systems Conference, October 2003 (to appear in Kluwer).

[20] Wilson S P, Kelly T P, McDermid J A. Safety Case Development: Current Practice, Future Prospects, Proc. 1st ENCRESS/12th CSR Workshop, September 1995, Springer-Verlag.

[21] DRIVE D11.4 – UML-HAZOP, D11.4, version 1.1, January 2003.